UNIVERSITY OF CENTRAL FLORIDA

DEPARTMENT OF INDUSTRIAL ENGINEERING AND MANAGEMENT SYSTEMS

p. 17

ANNUAL REPORT
ON
NASA GRANT NUMBER NAG2-625

Design of an Air Traffic Computer Simulation System to Support Investigation of Civil Tiltrotor Aircraft Operations

August 15, 1992

Ralph V. Rogers, Ph.D.
Principal Investigator

Department of Industrial Engineering and Management Systems
University of Central Florida
P.O. Box 25000
Orlando, FL
32816

INTRODUCTION

On 13 March 1990 the University of Central Florida formally accepted NASA Grant #NAG 2-625 for "Design of an Air Traffic Computer Simulation System to Support Investigation of Civil Tiltrotor Aircraft Operations." Full implementation of the research efforts was started in May 1990. The delay of the start of work was due to the arrival of funds in the middle of the academic semester. Both researchers and graduate students to be involved in the project were under contractual obligations until the end of the semester in May. Since May, 1990 work has proceeded steadily. This paper reports on the progress and accomplishments achieved under Grant #NAG 2-625 during the second year of efforts.

PROJECT DESCRIPTION

This research project addresses the need to provide an efficient and safe mechanism to investigate the effects and requirements of the tiltrotor aircraft's commercial operations on air transportation infrastructures, particularly air traffic control. The mechanism of choice is computer simulation. Unfortunately, the fundamental paradigms of the current air traffic control simulation models do not directly support the broad range of operational options and environments necessary to study tiltrotor operations. Modification of current air traffic simulation models to meet these requirements does not appear viable given the range and complexity of issues needing resolution. As a result, the investigation of systemic, infrastructure issues surrounding the effects of tiltrotor commercial operations requires new approaches to simulation modeling. These models should be based on perspectives and ideas closer to those associated with tiltrotor air traffic operations.

The principal objectives of the Tiltrotor Air Traffic Simulation System (TATSS) Project are to develop a design for computer software that will incorporate the following capabilities in an air traffic simulation model:

> Full freedom of movement for each aircraft object in the simulation model. Each aircraft object may follow any designated flight plan or flight path necessary as required by the experiment under consideration.

Object position precision up to ± 3 meters vertically and ± 15 meters horizontally.

Aircraft maneuvering in three space with the object position precision identified above.

Air traffic control operations and procedures.

Radar, communication, navaid, and landing aid performance.

Weather.

Ground obstructions and terrain.

Detection and recording of separation violations.

Measures of performance including deviations from flight plans, air space violations, air traffic control messages per aircraft, and traditional temporal based measures.

WORK ACCOMPLISHMENT

In the second year of effort the following tasks were accomplished:

1. Established basic object architecture for simulation model.

2. Identified data structure for implementing spatial blackboard (a.k.a. spatial template).

3. Identified intersection identification method.

4. Implemented prototype two-dimensional simulation model in MODSIM II based on basic object architecture and spatial blackboard data structure.

5. Began implementation of interface between MODSIM II and the expert system language CLIPS.

6. Began implementation of prototype three-dimensional simulation model in MODSIM II of basic object architecture.

Each of these tasks will be discussed in more detail in the following sections.

OBJECT ARCHITECTURE

The basic object model for the TATSS consists of two general types of objects with a total of five basic subtypes. The two general types are (1) Spatial Objects and (2) Model Management Objects. The Spatial Objects have two subtypes (a) Moving Objects and (b) Stationary Objects. The Model Management Objects consist of three subtypes: (a) the Spatial Template Object, (b) the Conflict Identifier Object, and (c) the Conflict Resolver Object. (Refer to Figure 1). In any given model there may be more than one instance of Moving and Stationary objects. However, there is generally only one instance per model of the Spatial Template, Conflict Identifier, and Conflict Resolver objects. Figure 2 provides an illustration of the interconnective relationship between these five types of objects. The general relationships between these objects are illustrated by following descriptive example.

DESCRIPTIVE EXAMPLE. A two dimensional model-space contains phenomena of interest. The phenomena consists of two dimensional entities and their behavior. Entities may have two basic behaviors; (1) they occupy space and (2) they may move from one location to another. Some entities may possess both behaviors (i.e. Moving Objects) while some may posses only the first (i.e. Stationary Objects). Stationary Objects are of random sizes and randomly assigned locations in the model-space. Moving Objects start at random locations within the model-space and move with a random velocity vector for a random period of time. Further, no two entities can occupy the same space at the same time.

In the basic TATSS object model the illustrative example presented would function as follows: All stationary objects report their dimensions and location in the model-space to the Spatial Template Object when they are instantiated. Moving Objects report their dimensions and their current positions in the model-space to the Spatial Template Object when they are instantiated. Before a Moving Object moves, it notifies the Spatial Template Object of its next desired position or location in the model-space. This establishes a planned model-space trajectory for the moving object which is represented graphically in the Spatial Template (Refer to Figure 3).

If the planned model-space trajectory of a Moving Object intersects the model-space trajectory of another Moving Object or the model-space of a stationary object then a possibility of a conflict exists. The Spatial Template Object identifies such intersections of the model-space trajectory polygon. These intersections represent potential conflicts. When intersections are identified, the Spatial Template notifies the Conflict Identifier Object that a possible conflict exists

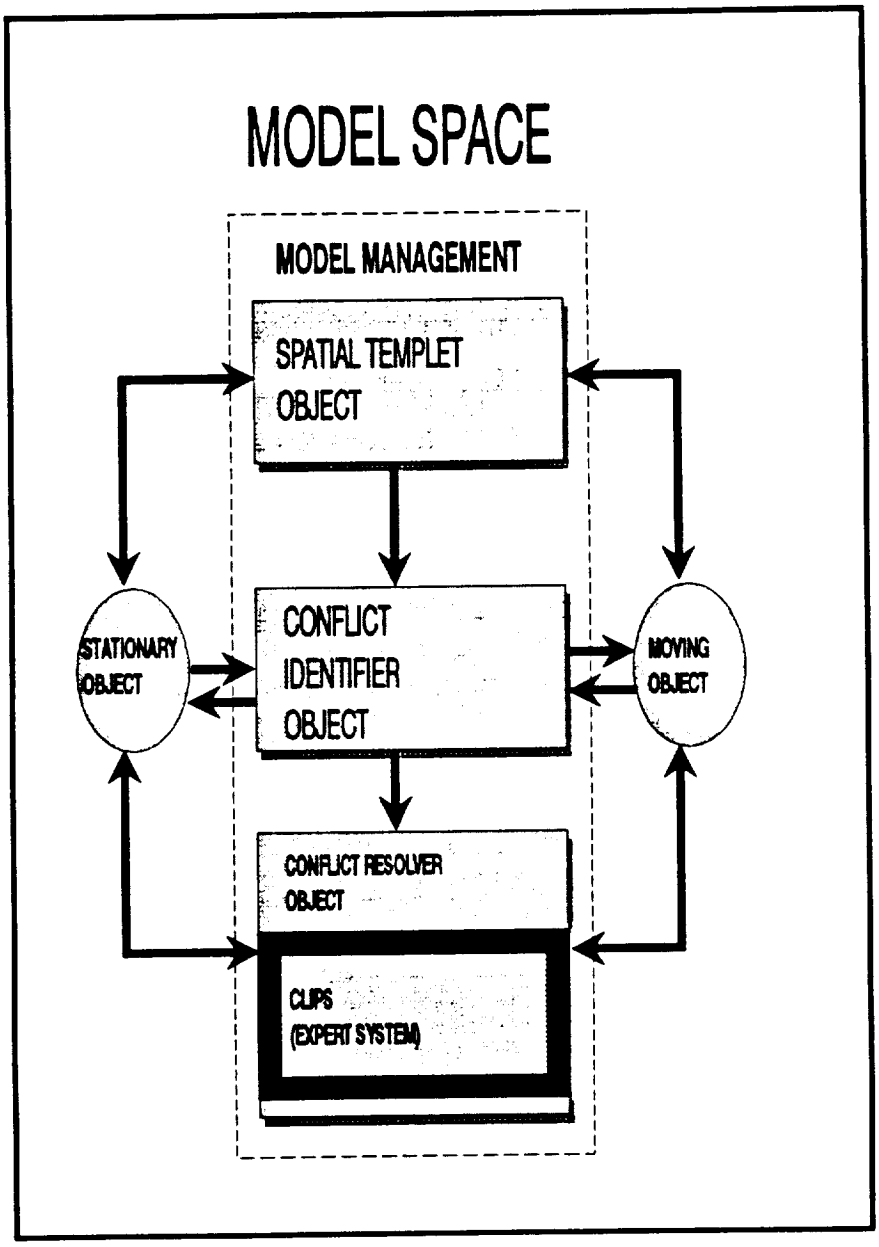

Figure 1. Interconnective relationship between basic object model type.

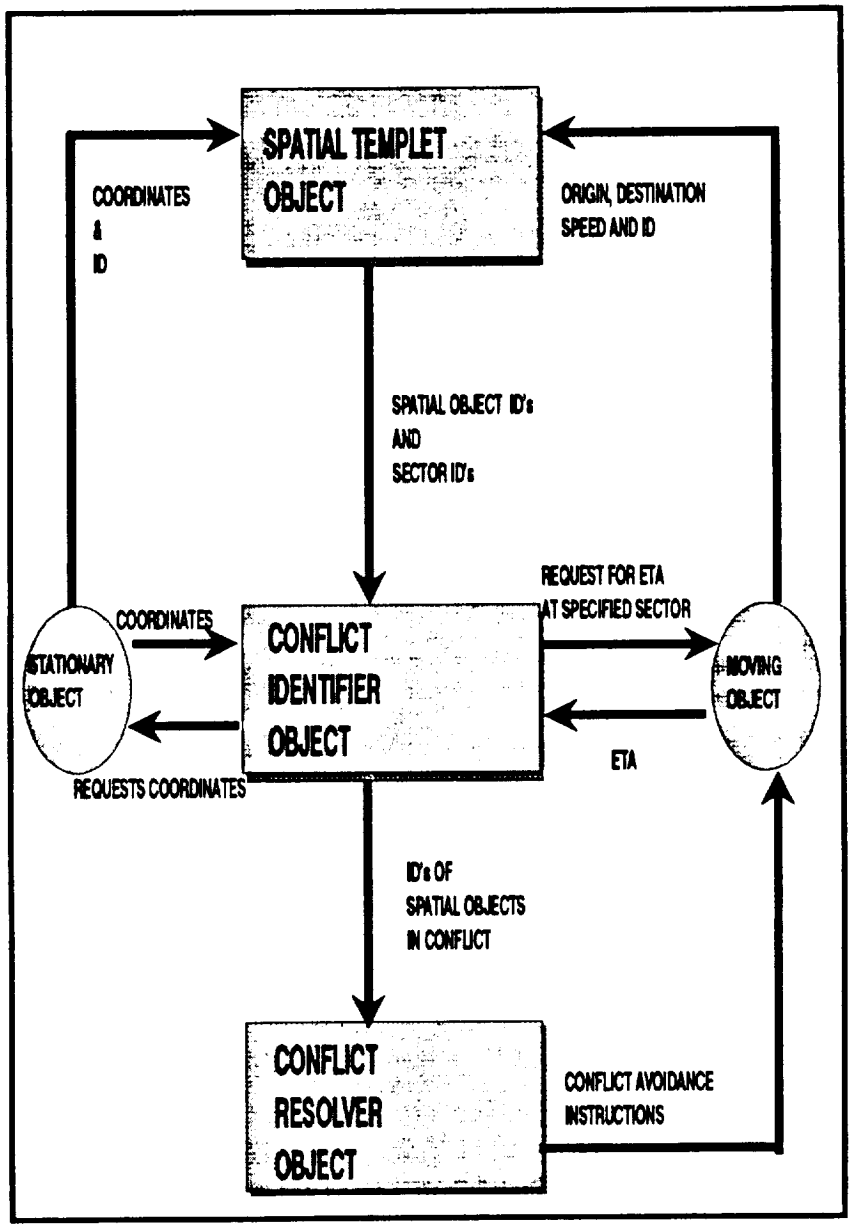

Figure 2. Basic object model types and messages.

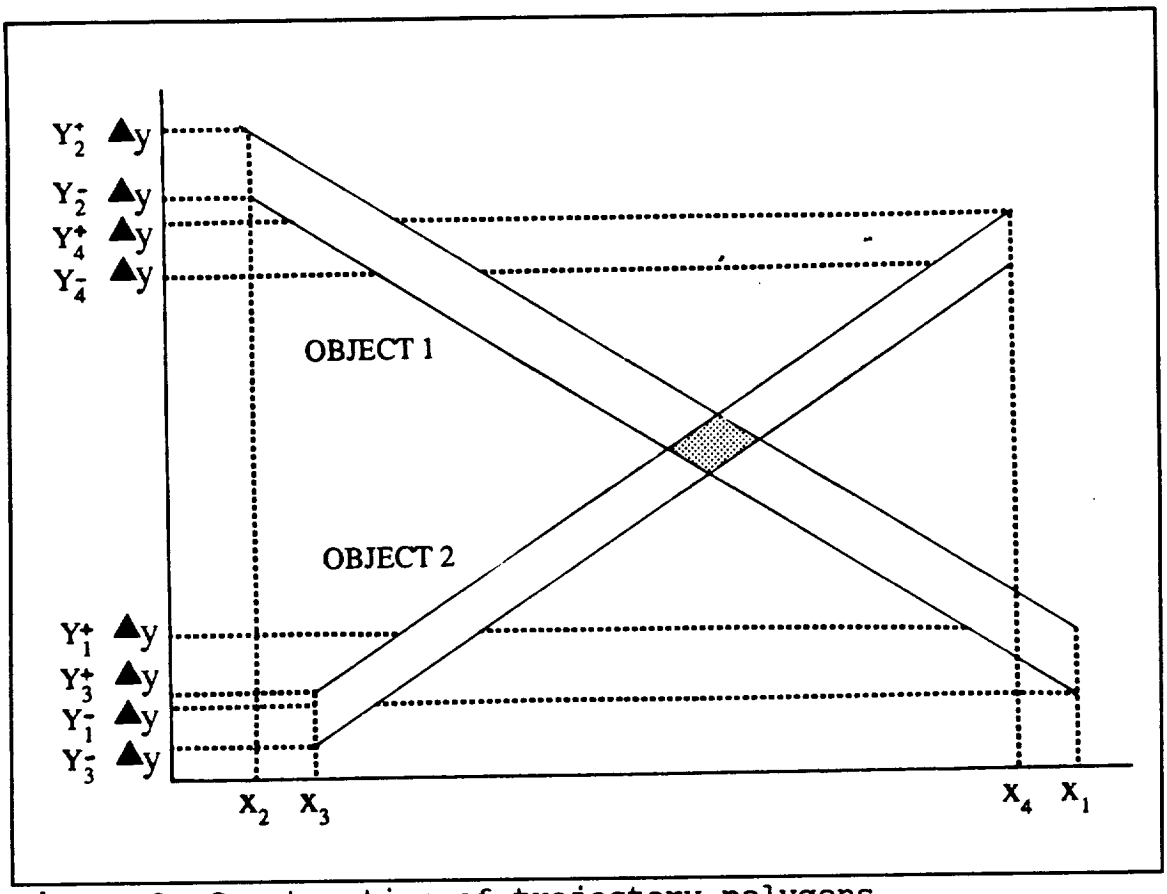

Figure 3. Construction of trajectory polygons.

between the two object instances in the model-space (Refer to Figure 4).

The Conflict Identifier then questions the two objects to determine when each object will arrive at the intersection neighborhood. Obviously for stationary objects, there is no arrival time. The objects are always there. If the arrival are separated by enough time, no actions are required and the moving object which wished to move to its next position is allowed to schedule its desired destination. The current position of the moving object and its desired/goal position establishes a spatial trajectory in the Spatial Template Object. If one of the two objects is a stationary object or if the separation time of two moving objects is insufficient, the Conflict Identifier Object notifies the Conflict Resolver Object that a conflict exists between two objects for model-space resources.

The Conflict Resolver determines the nature and extent of the conflict. The Conflict Resolver also determines the course action to be taken depending on the objects involved. For example, a conflict may be determined to exist between a

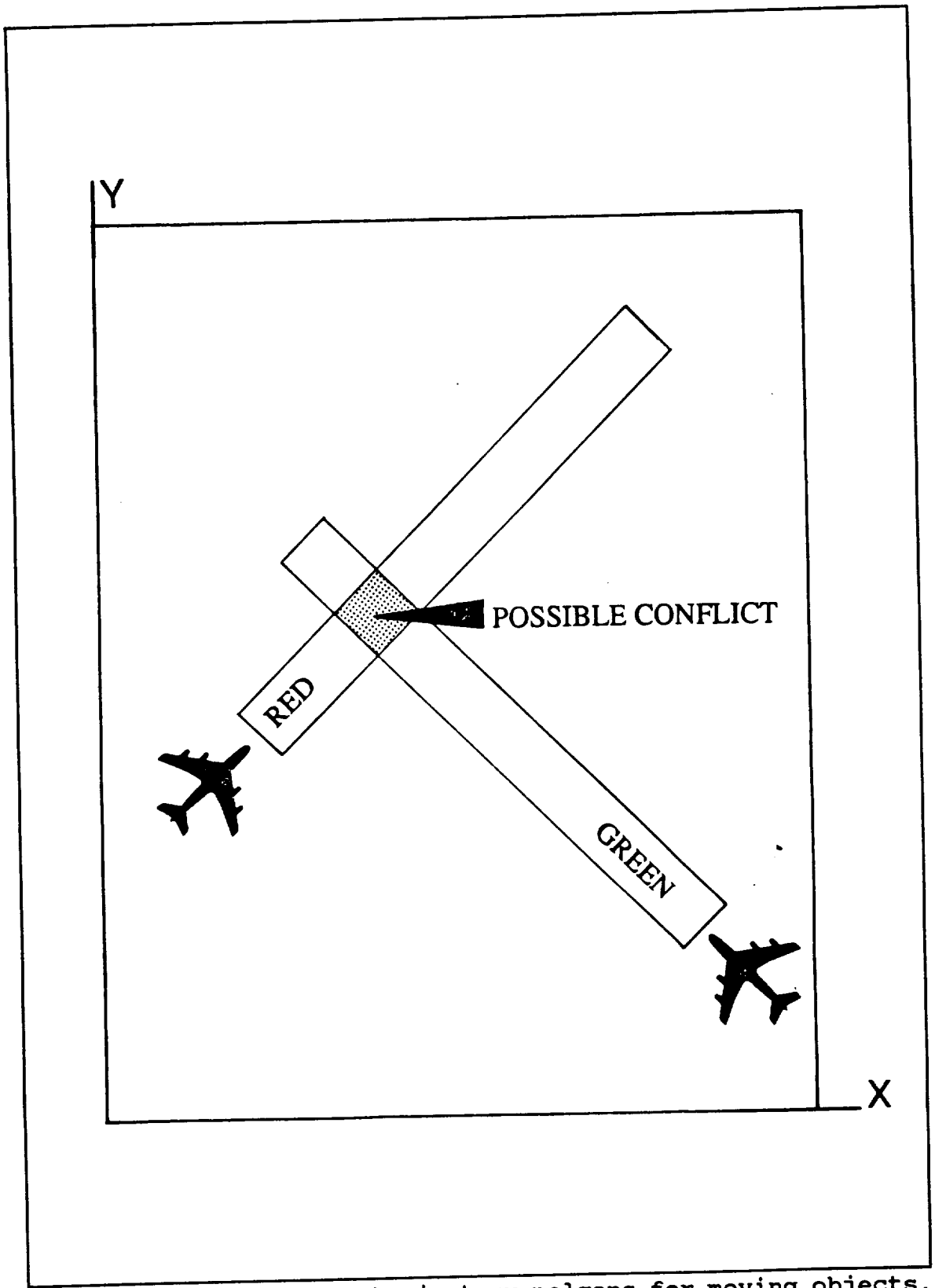

Figure 4. Model-space trajectory polgons for moving objects.

stationary object and a moving object. The conflict resolver object might simply notify the moving object that it will intersect a stationary object if corrective action is not taken. Or, the conflict resolver may determine the best course of action and issue new course trajectory to the moving objects. The choice would depend on the design of a particular model. Once a new course action is determined either by the resolver or the objects themselves, the new course objective is sent to the Spatial Template and the process begins again.

With this overview of the object architecture in mind, the following sections describe each of the object subtypes in more detail.

MOVING OBJECT. A Moving Object occupies space and follows a model-space trajectory as it moves from one location to another in model-space. A Moving Object is represented in the model as a geometric surface (i.e. a polygon) in either two or three dimensional space. A Moving Object may follow its own goal directed model-space trajectory based upon its own decision making capabilities and rules or it may be directed to follow specified trajectories by other objects (e.g. Conflict Resolver) in the model based on system model state. A Moving Object communicates with the Spatial Template Object, the Conflict Identifier Object, and the Conflict Resolver Object.

STATIONARY OBJECT. A Stationary Object occupies space and does not normally change locations in the model-space. A Stationary Object is represented in the model-space as a geometric surface (i.e. a polygon) in either two or three dimensional space. A Stationary Object communicates with the Spatial Template Object, the Conflict Identifier Object, and the Conflict Resolver Object.

SPATIAL TEMPLATE OBJECT. The Spatial Template Object provides a graphical memory of the model-space status between model events. The Spatial Template Object maintains this memory by a graphical representation based on the proposed model-space trajectory of a moving object or the occupied space of a stationary object. When new graphical descriptions are created representing model-space trajectories and model-space occupation, these new graphical objects are incorporated into the spatial template.

As each new graphical representation is added to the spatial template, the Spatial Template Object simply identifies instances when a new trajectory may compete for the same model-space resources as another object with a previously approved model-space trajectory. That is, it identifies when a potential conflict occurs between two or more model-space

objects. When potential conflicts are identified, the Spatial Template passes the identification of the involve objects to the Conflict Identifier Object.

There are two related principal technical issues associated with implementing the Spatial Template Objects of the TATSS base object architecture into an associated software model. The first technical issue is the identification of a data structure to represent the spatial template graphical based information. The second technical issue is how to detect when model-space object in the Spatial Template Object intersect. Each of these issues is addressed in more detail in subsequent sections of the report.

CONFLICT IDENTIFIER OBJECT. The Conflict Identifier Object receives from the Spatial Template Object the identification of spatial objects which have a potential conflict and the location in the model-space of that potential conflict. The Conflict Identifier Object sends messages to each of the potentially conflicting spatial objects asking them at what time each will arrive at the location of the potential conflict. If the difference in the arrival times of each is sufficiently large enough, then no conflict will occur. The spatial object whose new position change precipitated the conflict identification process is given permission to schedule its event past the location identified with the possible conflict. If the difference in the arrival times of each object is sufficiently small enough, then a conflict will occur. The Conflict Identifier Object then sends a message to the Conflict Resolver Object containing the identification of each of the conflicting spatial objects, the location in the model-space of the conflict, and the time of the conflict.

CONFLICT RESOLVER OBJECT. The Conflict Resolver Object determines the actions specified by the system model designed to resolve the conflict. After determining the appropriate course of actions, the Conflict Resolver Object sends message(s) to one or both affected spatial object instructing them on actions to be taken to resolve the conflict. If the system model dictates decision making by the one or both objects to resolve the conflict, the Conflict Resolver Object informs one or both objects of the conflict and the other object involved. In such a case, each spatial object must be designed to make conflict resolution decisions.

If the system model dictates that determination of the conflict resolution actions be made by the Conflict Resolver Object, the conflict resolution mechanism must be directly or indirectly a part of the Conflict Resolver Object. Once the appropriate actions are determined by the Conflict Resolver Object, these actions are sent as messages to the involved

objects. Each spatial object must be so designed to implement these specified actions.

The conflict resolver mechanism is best thought of as rule-based approach to conflict resolution. This mechanism may include hard coded software methods and procedures as part of the system software model or may be based on rule-based inferencing through a separate knowledge base typical of expert-system approaches. To obtain this latter capability, the Conflict Resolver Object must include or interface with an expert system shell or language. In TATSS this expert system shell is CLIPS. CLIPS is a C based expert system language and environment which can be interfaced with MODSIM II. A more complete description of this interface will be presented in a subsequent section.

SPATIAL TEMPLATE DATA STRUCTURE

In previous discussions, the Spatial Template was identified as a graphical representation of the model state space. The Spatial Template is both more and less than this. The interest is fundamentally the identification of the **POTENTIAL** conflicts. That is, to identify those objects whose model space trajectories may compete for the same model space resources (e.g. space) at the same time. The goal is reduce the message traffic between model objects necessary to determine if conflicts will occur. The approach taken is to graphically represent the model space trajectory of model system objects and to identify the intersection of trajectories before allowing the object to proceed. How to identify these potential conflicts in an efficient manner is dependent on the how to represent the model state space information.

TATSS's approach taken to address these issues requires a partitioning of the model space. Currently, the model space is divided into equal sectors. Thus, the model space is represented as a either a two or three dimensional cartesian coordinate system. Sectors are identified by their coordinate numbers (e.g. sector 1,3, 2). The model space trajectory of an object represented by its associated polygon is position with reference to this coordinate system (Refer to Figure 5). The sectors through which the trajectory polygon goes are identified. Associated with each sector is a list of objects whose trajectory is schedule to go through some part or all of that sector. When new object trajectories are added a the sector list, a check is made to determine if any other object trajectories are also associated with that sectors. If an object trajectory is already associated with a sector, then a potential conflict exists. The Conflict Identifier is then

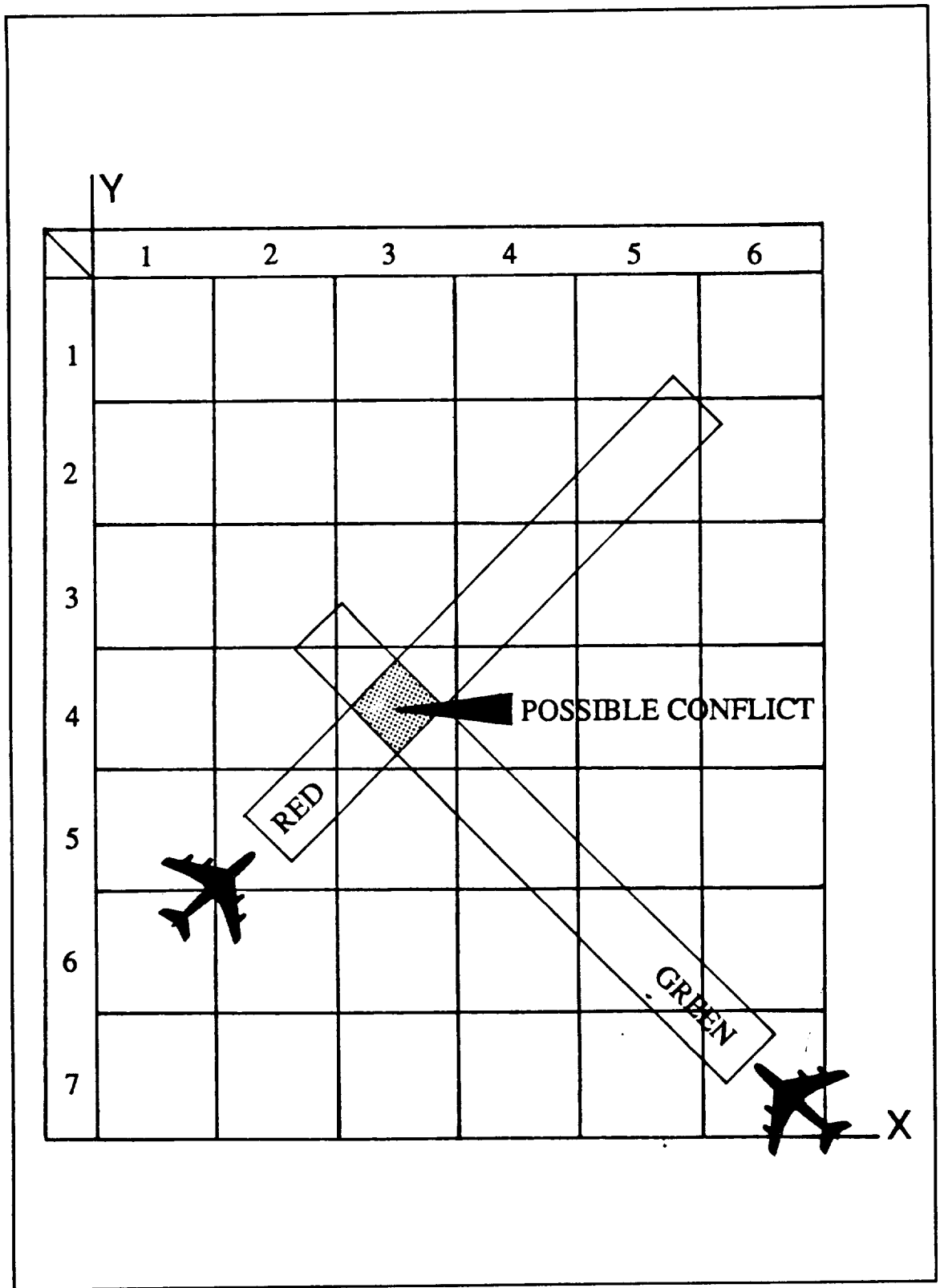

Figure 5. Model-space trajectory polygons in coordinate reference system.

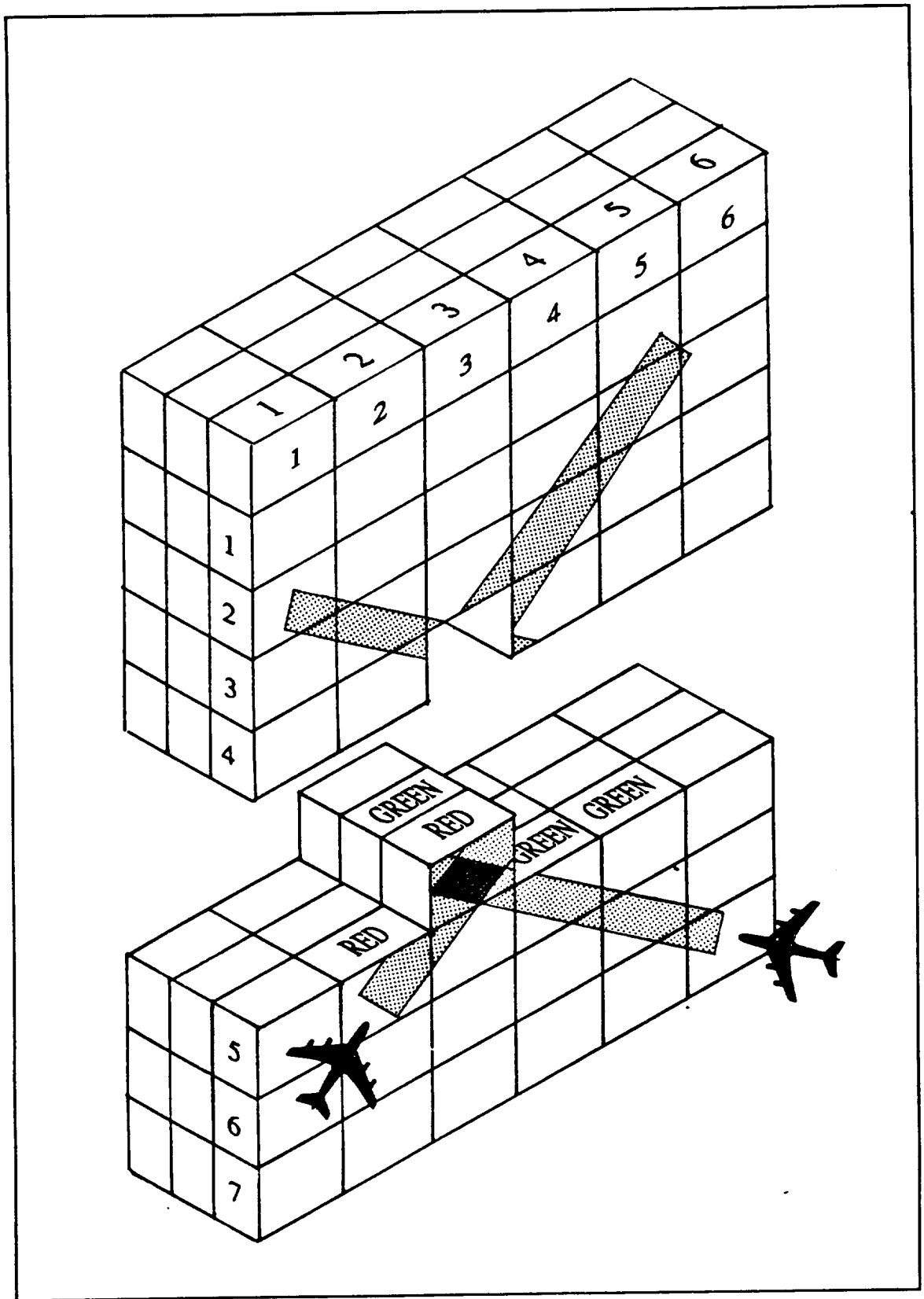

Figure 6. Spatial Template with sector queues.

notified and the names of the object with potential conflicts are sent with the notification.

The data structure employed to capture this approach is a dynamic array of queues. The sectors of the model space become elements in the array. Each element in the array is a queue. As object trajectories cross a sector, the name of the object is added to the sector (i.e. queue) (Refer to Figure 6). Once a new model trajectory has been established for an object, the name of the object is remove from the sectors associated with the old model trajectory.

In the object-oriented software implementation, each sector element is defined as a queue object. The use of objects to describe sectors enables the exploitation of the dynamic array. The dynamic array will only use the memory necessary for the active sector objects. Thus, if no sectors have objects associated with them, they are not created, and do not require computational resources. Likewise, as a sector becomes empty, that sector object may be disposed of, freeing computational resources.

INTERSECTION IDENTIFICATION

The identification of the intersection of the model space trajectory of objects follows a hierarchial decision process. The first level in the process determines if the model space trajectories (i.e. the trajectory polygons) of objects transverse or occupy the same sector or sectors. If more than one model space trajectory intersects a sector, then the name of the objects associated with that sector are sent to the conflict identifier to establish if a conflict actually exists. The issue of concern is to identify what sectors an object model space trajectory intersects. This section discusses the approach used in TATSS to identify these sectors intersected by an object's model space trajectory.

When an object reports its current and goal positions to the Spatial Template, the Spatial Template defines the trajectory polygon in the model space. Determination of which sectors are intersected is made by scanning along the x direction of each side of two parallel sides of the polygon. The two parallel sides chosen are the two which are parallel to the velocity vector. Scanning is made in the direction of the x component of the velocity vector. Scanning along the other sides of the polygon is not necessary because the width of the polygons are less than one sector width.

As the scanning occurs along the polygon side, the contents of each sector queue intersected is checked. If another object's (or objects') trajectory has already been recorded in the

queue, the name of the object(s) along with object who is scanning is sent to the conflict identifier for further conflict evaluation.

Scanning is performed by establishing the coordinate border values for the sector under consideration. For example, in a two dimensional coordinate system, one side of a polygon is defined by the two points (1,2) and (8,11). Movement is in the direction from (1,2) to (8,11). Sectors are 3 unit square. Sector 1 would be defined by sector borders of X=3 and Y=3. The point (1,2) is less than the value for each border but greater than 0. Therefore, point (1,2) is in the first sector. Scanning continues along the line defined by the points (1,2) and (8,11). The next test point is X border value of 6 and the Y value where the polygon side intersect the x axis with value 6. The scan determines what sectors the line intersects/crosses between this new point and the old sector value (i.e. sector 1, 0<X<3, 0<Y<3). The sector scan continues until the sector contains the end points of the polygon is identified or an identified conflict results in a new trajectory plan.

The area of sector identification for trajectory polygons is an area which requires further investigation. More efficient and faster approaches than the one used should be available or if not then developed. This is the one area of the project which would benefit from more intensive investigation.

MODSIM II AND CLIPS INTERFACE LINK

The interface link between ModSim II and CLIPS was implemented on an UNIX system V, using NASA's CLIPS version 5.0 for the expert system, and ModSim II version 1.6 for the simulation system. CLIPS and MODSIM II were run as separate processes in UNIX with no parent child relationship required. The interface link allows messages of any type and length to be exchanged between the CLIPS processes and the ModSim II processes. The interface link default as implemented allows eight separate processes to communicate.

The prototype communications between the simulation model's Conflict Resolver Object in ModSim II and the Resolution Expert in CLIPS were messages about model object trajectory conflicts and solutions for those conflicts. Messages from the Conflict Resolution Object concerned Moving Objects in the simulation identified as in conflict. Messages from the Resolution Expert to the Conflict Resolution Object provided actions commands for the Moving Object necessary to avoid the identified conflict.

The interface link was implemented in the C language using UNIX System V Inter Process Communications (IPC). The IPC method used was a System V FIFO with 2-way FIFO's used between separate processes. The interface link opens the FIFO communication channel with a mknode call to the UNIX system. The FIFO method was chosen to preclude the UNIX problem of concurrent read and write by separate processes. The majority of the interface link code is contained in a separate C header file. This single separate header file must be compile both with the ModSim II simulation and the CLIPS Resolution Expert program. The violation of C program conventions to include code in a *.h file was done to allow a generic interface.

CLIPS and ModSim employ the interface link through two separate simple functions. The two functions are send() and receive(). The UNIX methods invoked are completely hidden from users of the interface link. The advantage of this approach is that it allows the IPC method currently used to be modified as the needs of CLIPS or ModSim interface may require. For example, if faster communication is desired the interface link header file could be modified to use shared memory. CLIPS and ModSim both include this common header file so recompiling the programs with the new interface header file is all that is required. The ease of modification was judged to be of more value than the C language guideline regarding *.h files. The send and receive functions were included in CLIPS by adding them to the usr_define section of main.c, including the interface link file and recompiling the entire CLIPS library. Example of the procedure is detailed in the CLIPS Advanced Programmers Manual.

The send and receive functions were included in ModSim by adding a separate file that included interface link file and had send and receive defined as NONMODSIM. The ModSim II Reference Manual provides the details and an example of this type of procedure.

TWO DIMENSIONAL TATSS PROTOTYPE

A two dimensional prototype of TATSS was implemented. The prototype models a 150 kilometer by 150 kilometer two dimensional space with four moving objects and two stationary objects. Within the space, the four moving objects are randomly assigned to some starting location. A destination point is randomly selected for each moving object. Likewise, a speed is randomly assigned for each moving object. This results in an arrival event being defined for each object. After arrival the destination point, a new destination point is randomly assigned as well as a new speed. This four moving objects continue to randomly move throughout the model space during the duration of the simulation.

The TATSS system manages the movement of the moving objects to prevent collisions between moving objects and between moving objects and stationary objects. A potential conflict is defined to be when objects are scheduled to occupy the same sector within five minutes of each other. A conflict is defined to be when the intersection of two objects' model space trajectories occur within five minutes of each other.

Conflicts are resolved between moving objects by having the objects with the smallest identification number wait until the other object moves far enough to eliminate the conflict. Conflicts between moving objects and stationary objects are resolved by having the moving object randomly selecting a different destination. Conflict decisions are made by the CLIPS expert system module of TATSS. While the rules used in the prototype are simple and basic, they do demonstrate the feasibility of the approach.

The TATSS prototype has been exercised using the model described above. During these exercises TATSS was able to recognize conflicts and implement actions to resolve those conflicts. In our testing, moving objects where able to avoid collisions with other moving objects and with stationary objects.

PLANNED EFFORT

The project has to date accomplished its major goals. Work is continuing on development of the three dimensional spatial template. This work should be completed by the end of August, 1992. A final report will be prepared and is scheduled for delivery in mid-October, 1992.

One paper was presented at the 1991 IEEE/SMC International Conference on Systems, Man, and Cybernetics October 13-16, 1991 in Charlottesville, Virginia. The title of the paper was, "Understated Implications of Object-Oriented Simulation and Modeling."